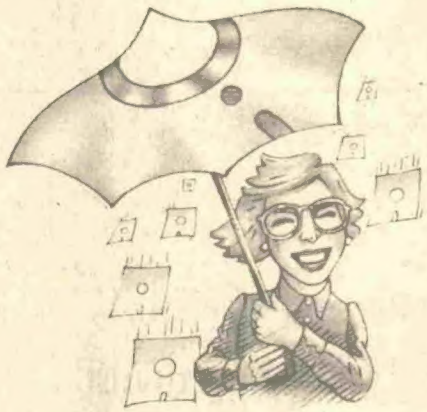
ยืน ภู่วรวรรณ

พัฒนาการคอมพายเลอร์ คลิปเปอร์ 5.0

เสนอโครงสร้างพื้นฐานของโปรแกรมที่เขียนด้วยคลิปเปอร์เวอร์ชัน 5.0 มีอะไรเด่น ๆ บ้างล่ะ



ในช่วงกว่าสิบปีที่ดีเบสมีพัฒนาการมา การใช้งานจัดการฐานข้อมูลบนพีซีเป็นที่แพร่หลายกว้างขวางมากมาย ดีเบสเป็นโปรแกรมที่มีไวยากรณ์ของตัวเอง ที่นำมาเขียนเป็นโปรแกรมสั่งงานได้ หลังจากที่ดีเบสประสบผลสำเร็จก็มีโปรแกรมอื่นตามมาอีกหลายโปรแกรม ในบรรดาโปรแกรมที่พัฒนาขึ้นภายหลังและมีจุดเด่นในเรื่องของการเป็นคอมพายเลอร์เห็นจะได้แก่ คลิปเปอร์

คลิปเปอร์เป็นโปรแกรมที่พัฒนาโดยบริษัท แนนทัคเคต ที่มีบทบาทและพัฒนาให้ดีขึ้นเป็นลำดับ แนนทัคเคตเริ่มพัฒนาคลิปเปอร์ตั้งแต่ปี 2528 และปรับปรุงให้ดีขึ้นเรื่อย ๆ จนในที่สุดมีการกำหนดรูปแบบและไวยากรณ์ โดยเพิ่มคำสั่งอื่น ๆ เข้ามาให้มีลักษณะเป็นภาษาคอมพิวเตอร์ภาษาหนึ่ง มีโครงสร้างการเขียนคล้ายภาษาโครงสร้างทั่วไป เช่น ภาษาซี พัฒนาการของคลิปเปอร์ในเวอร์ชัน 5 จึงเป็นสิ่งที่มีประโยชน์และน่าสนใจ มุมดีเบสจึงเสนอรายละเอียดเทคนิควิธีการของคลิปเปอร์ โดยจะใช้เวอร์ชัน 5 เป็นหลัก

เริ่มจากการใช้คอมพายเลอร์

ดีเบสเป็นโปรแกรมอินเตอร์พรีตเตอร์ การทำงานของดีเบสเริ่มจากการใช้คำสั่ง DO เพื่อเรียกแฟ้มโปรแกรมเข้ามาทำงาน แต่สำหรับคลิปเปอร์เป็นคอมพายเลอร์ที่จะแปลงแฟ้มโปรแกรมต้นฉบับที่เขียนเป็นแฟ้มเอกสาร มาแปลงเป็นออปเจ็กต์และลิงค์กับไลบรารีเพื่อให้ได้แฟ้มเอ็กซีคิวต์ สามารถเรียกทำงานได้ทุกอย่างเสมือนเป็นโปรแกรมประยุกต์โดยไม่ต้องมีรันไทม์อยู่ด้วย

คลิปเปอร์จึงเป็นคอมพายเลอร์เหมือนคอมพายเลอร์ภาษาคอมพิวเตอร์อื่น ๆ เช่น ภาษาซี ภาษาปาสคาล ฟอร์แทรน โคบอล ฯลฯ การคอมพายล์โปรแกรมด้วยคลิปเปอร์เริ่มจากการเรียกแฟ้มด้วยคำสั่ง clipper เช่น เมื่อต้องการคอมพายล์โปรแกรม payroll ซึ่งเขียนเป็นแฟ้มต้นฉบับและเก็บไว้ในชื่อแฟ้มว่า payroll.prg แล้วใช้คำสั่งดังนี้

```
Clipper payroll
```

หากโปรแกรม payroll มีการเรียกโปรแกรมอื่นที่เก็บไว้ในแฟ้มที่แยกจากแฟ้มต้นฉบับ เช่น โปรแกรมเรียก tax.prg และ report.prg โดยภายใน payroll มีคำสั่ง do tax และ do report คลิปเปอร์จะเรียกโปรแกรมเหล่านี้เข้ามาคอมพายล์ร่วมด้วย

ทำนองเดียวกันถ้าหากใช้คำสั่ง SET PROCEDURE คลิปเปอร์ก็จะเรียกแฟ้มนั้นเข้ามาคอมพายล์ร่วมด้วย โดยได้ออปเจ็กต์ชุดเดียวกับโปรแกรมที่ถูกเรียกโปรแกรมแรก

บางครั้งต้องการคอมพายล์เฉพาะแฟ้มใดแฟ้มหนึ่ง โดยไม่ต้องให้เรียกแฟ้มอื่นที่เรียกทำงานโดยลำพัง DO มาคอมพายล์ร่วมด้วย การสั่งการเป็นการใช้สวิตช์เป็นตัวบอก เช่น ต้องการคอมพายล์เฉพาะ payroll.prg ก็สั่งได้ดังนี้

```
Clipper payroll /m
```

การใช้พารามิเตอร์สวิตช์เป็น /m หมายถึง ให้คอมพายล์เฉพาะแฟ้ม และไม่ต้องเรียกแฟ้มอื่นที่เกี่ยวข้องด้วยมาคอมพายล์ แต่จะต้องแยกคอมพายล์แฟ้มอื่นให้ได้ .obj แล้วจึงนำมาเชื่อมกันใหม่ด้วยการลิงค์

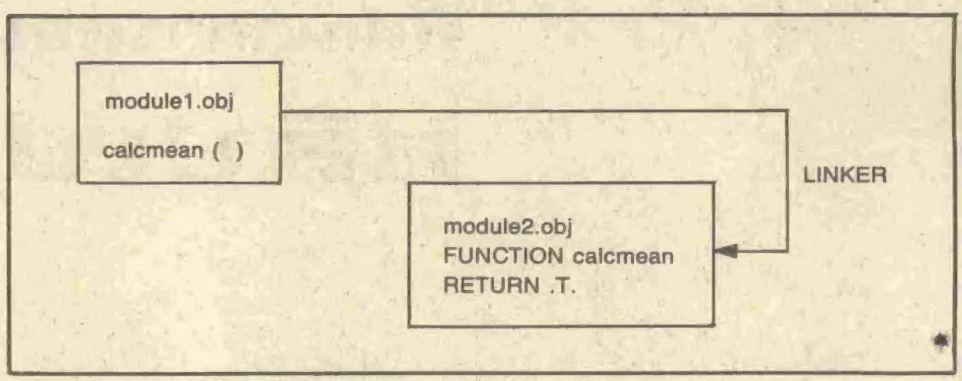
หากต้องการรวมแฟ้ม .prg หลายๆ แฟ้มเข้าหากัน แล้วคอมพายล์ให้ได้ .obj ตัวเดียว เราจะตั้งแฟ้มให้มีส่วนขยาย .clp โดยเป็นแฟ้มเอกสารเก็บรายละเอียดแฟ้มที่ต้องการคอมพายล์เอาไว้ เช่น แฟ้ม procl.clp มีรายละเอียดข้อมูลที่ต้องการให้คอมพายล์แฟ้ม tax กับ report ก็จะมีข้อมูลสองบรรทัดดังนี้

```
tax
report
```

เมื่อสั่งคอมพายล์ก็ใช้คำสั่งดังนี้

```
clipper @procl
```

กรณีนี้จะสร้างออปเจ็กต์ที่ชื่อ procl.obj ที่มีรหัสออปเจ็กต์ของทั้งสองแฟ้มนั้นอยู่รวมกัน



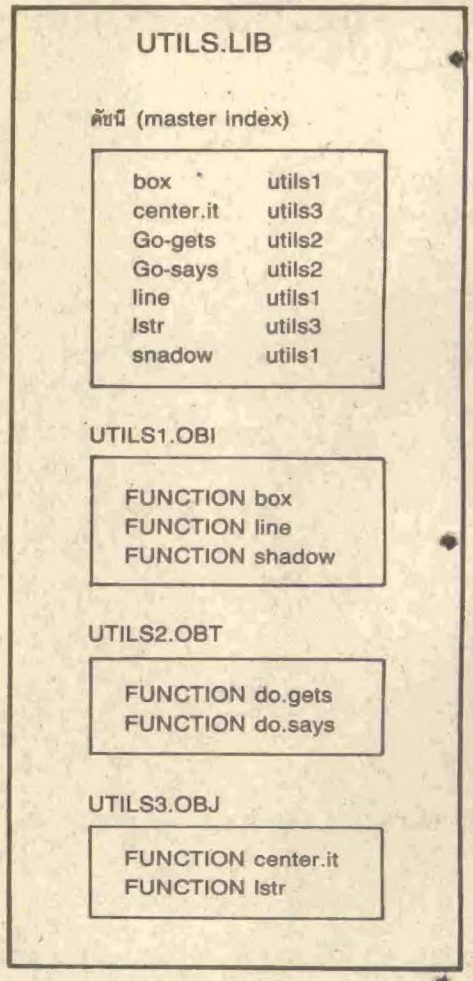
รูปที่ 1 การลิงค์เชื่อมระหว่างโมดูลของ .obj

การลิงค์หรือเชื่อมต่อระหว่างโมดูลออปเจ็กต์

คลิปเปอร์ 5.0 ใช้ลิงค์จากโปรแกรม RTLINK ความจริงแล้วจะใช้ LINK ของไมโครซอฟต์ก็ได้ RTLINK เป็นลิงค์ที่มีข้อเด่นและดีกว่า LINK ของไมโครซอฟต์ เพราะสามารถลิงค์เพื่อทำโอเวอร์เลย์ ซึ่งทำได้ทั้งแบบสแตติกและไดนามิก และสามารถเชื่อมต่อกับภาษาซีและแอสเซมบลีได้

การลิงค์เป็นการเชื่อมโยงฟังก์ชันหรือโปรแกรมย่อย (ในคลิปเปอร์ใช้เป็นฟังก์ชันเหมือนภาษาซี) ที่อยู่ต่างโมดูลกันเข้าหากัน ในการคอมพายล์ฟังก์ชันที่ไม่มีอยู่ในไฟล์ที่คอมพายล์ขณะนั้น คอมพายเลอร์จะสร้างตารางเรียกฟังก์ชันเป็นแบบ external หรือกล่าวอีกนัยหนึ่งคือ ถือว่าเป็นฟังก์ชันนั้นอยู่ภายนอกจะเชื่อมโยงขณะลิงค์ ดังตัวอย่างที่แสดงรูปที่ 1

หากการลิงค์ไม่สามารถหาฟังก์ชันที่จะเชื่อมโยงได้ ก็จะเกิดข้อผิดพลาด โดยปกติคลิปเปอร์จะมีไลบรารีให้ไว้ การเชื่อมโยงจึงต้องทำกับไลบรารีที่เก็บออปเจ็กต์ที่ต้องใช้ไว้ด้วย ไลบรารีจะเป็นเสมือนแฟ้มที่เก็บ .obj ไว้หลาย ๆ ตัว โดยมีส่วนดัชนีที่หัวแฟ้มบอกว่าในไลบรารีนี้เก็บอะไรไว้บ้าง เช่นรูปที่ 2 เป็น



รูปที่ 2 โครงสร้างไลบรารี

ตัวอย่างของไลบรารีที่แสดงเพื่อความเข้าใจคือ UTILS.LIB ภายในจะมีโมดูลของออปเจ็กต์ 3 โมดูล แต่ละโมดูลจะมีหลายฟังก์ชัน ภายในไลบรารีจะมีดัชนีบอกว่าแต่ละฟังก์ชันอยู่ในโมดูลชื่ออะไร

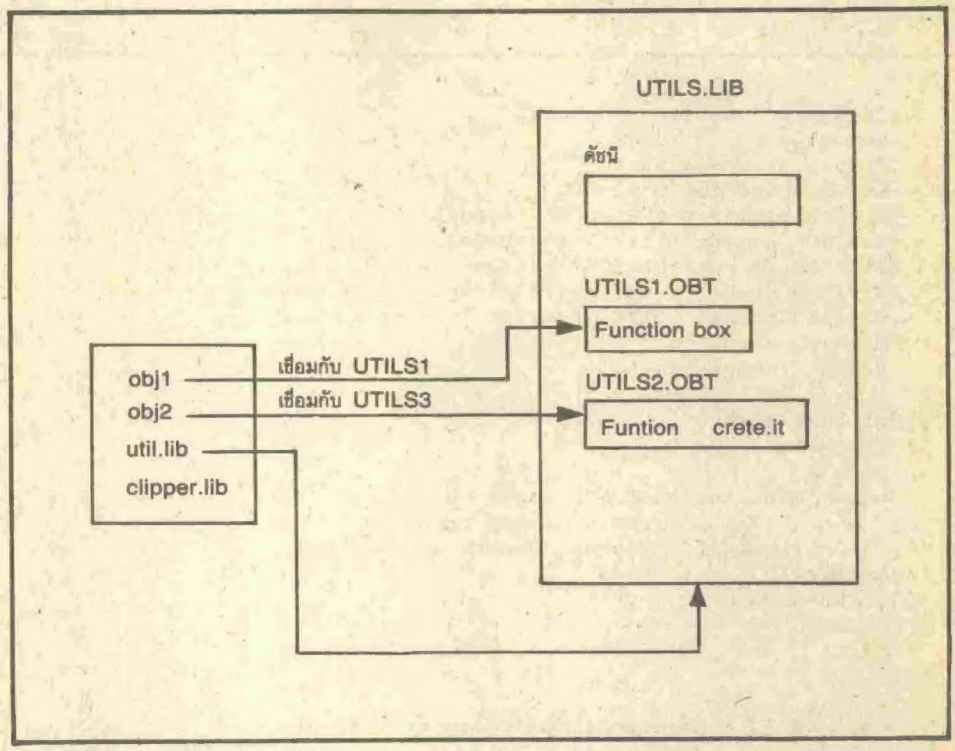
สมมติว่ามีโปรแกรมประยุกต์ที่ผ่านการคอมพายล์และจะต้องลิงค์เข้าหากันชื่อ obj1.obj และ obj2.obj โปรแกรมทั้งสองนี้เมื่อเชื่อมกันจะได้ .exe ที่ต้องการ แต่ obj1.obj มีเรียกฟังก์ชัน center.it ดังนั้นจะต้องลิงค์เชื่อมกับ UTILS.LIB อย่างไรก็ดี โปรแกรมที่เขียนขึ้นจะต้องลิงค์กับ clipper.lib เสมอ เพราะฟังก์ชันของโปรแกรม clipper.lib เป็นฟังก์ชันพื้นฐานหลายอย่างที่ต้องเรียกใช้อยู่แล้ว ดังนั้นการเชื่อมโยงของโปรแกรมจะเป็นดังรูปที่ 3

จะเห็นว่าการลิงค์จึงเป็นการเชื่อมโยงระหว่างฟังก์ชันต่าง ๆ ที่เรียกใช้ ทั้งนี้เพราะการเชื่อมโยงที่เกิดขึ้นจะเกิดขึ้นทั้งจากโปรแกรมที่เขียนในส่วนของโมดูล แยกจากกันหรือเชื่อมโยงเข้ากับส่วนของฟังก์ชันย่อยที่อยู่ในไลบรารี ส่วนของไลบรารีจะเป็นไลบรารีที่คลิปเปอร์กำหนดให้แล้วหรือผู้ใช้สร้างฟังก์ชันของตัวเองแล้วเก็บไว้ในไลบรารีก็ได้

ในการใช้ RTLINK ผู้ใช้สามารถใช้โหมดตอบโต้โดยโปรแกรมลิงค์จะถามถึงออปเจ็กต์ว่าจะให้ใช้แฟ้มอะไรบ้าง และไลบรารีจากที่ใดหรืออาจจะเรียกลิงค์แบบคำสั่งบรรทัดเดียว เช่น ต้องการลิงค์ test.obj โดยลิงค์กับ clipper.lib ซึ่งเป็นดีฟอลต์อยู่แล้ว เราจะใช้คำสั่งดังนี้

```
RTLINK test, test, test, clipper
```

– test ตัวแรกเป็นชื่อออปเจ็กต์ไฟล์



รูปที่ 3 การเชื่อมโยงกับไลบรารี

รูปที่ 4 ฟังก์ชัน inputdata ( )

```
/* TESTDEL.PRG ---test function dele_rec, rblank and add_rec
                  to use in database file */
// Manifest constant definition
#include "box.ch"
#define ONE 1
#define TWO 2
#define THREE 3
// ------------------------------------------------------------
local nchoice
use person
set index to perid   && INDEX KEY : ID
clear
do while .t.
   nchoice := mainmenu()
   do case
      case nchoice = ONE
           listdata()
      case nchoice = TWO
           dele_rec()
      case nchoice = THREE
           add_rec()
       otherwise
            exit
   endcase
clear
enddo
clear
close all
return
```

```
FUNCTION mainmenu(menuchoice)
   set wrap on
   @ 23,1 clear to 23,80
   set message to 23 center
   @ 5,16 ,18,29 box B_DOUBLE_SINGLE
   @ 8,20  prompt "LIST" message "list data"
   @ 10,20 prompt "DELETE" message "delete record"
   @ 12,20 prompt "ADD" message "add new record"
   @ 14,20 prompt "QUIT" message "return to dos"
   menu to menuchoice
   return (menuchoice)

FUNCTION listdata
IF len(alias()) = 0
   clear
   wait 'Function LISTDATA asked to ' +;
          'list all record in an open '+;
          'buffer.  Program aborting.' to junk
   CLOSE ALL
   CLEAR ALL
   CLEAR
   CANCEL
ELSE
    clear
    disp all id,name,position,bdate,status,salary
    ?
    wait 'PRESS ANY KEY TO CONTINUE'
ENDIF
RETURN .t.

FUNCTION dele_rec
PRIVATE rvalue
*** function blanks out all data in a record and
*** then sets the  delete flag to TRUE
num =0
IF len(alias()) = 0 .or. reccount() = 0
   clear
   wait 'Function DELE_REC asked to ' +;
          'delete a record in an open '+;
          'buffer.  Program aborting.' to junk
   CLOSE ALL
   CLEAR ALL
   CLEAR
   CANCEL
ELSE
   @ 10,26 clear to 10,80
   @ 9,26,11,63 box B_DOUBLE_SINGLE
   @ 10,28 say  'ENTER RECORD NUMBER TO DELETE' get num picture '99'
   read
   go  num
   rvalue = rblank()     && important function!
   IF rvalue             && if record was blanked
      DELETE             && delete it
   ENDIF
ENDIF
RETURN(rvalue)

FUNCTION rblank
PRIVATE f_name, i, rvalue
***  Blanks out all data in all fields of
***  the current record of the currently selected
***  data table
rvalue = .t.
```

– test ตัวที่สองเป็นเอาต์พุตไฟล์ คือไฟล์ที่จะเรียกรันได้ (exe)

– test ตัวที่สามเป็นแม็พไฟล์

– clipper คือไลบรารี

เราสามารถจะ test ตัวที่ 2 และ 3 ออก ถ้าใช้ชื่อเหมือนออปเจ็กต์ไฟล์ได้ ดังนี้

```
RTLINT test,,,clipper
```

หรือจะใช้ให้เต็มที่โดยใช้คำสั่งนี้

```
RTLINK FILE test OUTPUT
test1 LIBRARY utils
```

ในที่นี้ RTLINK จะใช้คีย์เวิร์ด ที่สำคัญคือ FILE OUTPUT MAP และ LIBRARY ในคำสั่ง

ตัวอย่าง โปรแกรมคลิปเปอร์ 5.0

โปรแกรมที่เขียนตามไวยากรณ์ของคลิปเปอร์ใช้รูปแบบของภาษาซีอ มาก ซึ่งจะขออธิบายรายละเอียดใ ตอนต่อ ๆ ไป เพื่อให้เห็นโครงสร้า ของโปรแกรม จึงเขียนโปรแกรมตัวอย่า เพื่อใช้ในการแสดงเมนูบนจอภาพ โด เมนูมีสี่ข้อคือ LIST DELETE ADD และ QUIT กับแฟ้มข้อมูลที่เปิดไว้หนึ่ แฟ้ม โดยที่แฟ้มที่เปิดนี้คือ perso แฟ้มดัชนีตามคีย์ perid โดยขออธิบา ตามฟังก์ชันที่เขียนดังนี้

ส่วนหัวโปรแกรม

โปรแกรมที่เขียนด้วยคลิปเปอ ใช้คอมเมนต์ด้วย /* หรือ // ส่วนห ของโปรแกรมคือ # include และ define เช่น # include "box.ch" คื เรียกแฟ้ม box.ch มาคอมพายล์ ซึ่งเป็ การกำหนดรูปแบบของการตีกรอบตา มาตรฐานจอ รายละเอียดส่วนนี้จะเห็

```
   i = 1
   DO WHILE len(field(i)) # 0      && for all fields
      f_name = field(i)            && field name
      DO CASE
         CASE type(f_name) = 'C'  && character data
              REPLACE &f_name with ''
         CASE type(f_name) = 'N' .or. ;
              type(f_name) = 'F'  && numeric data
              REPLACE &f_name with 0
         CASE type(f_name) = 'D'  && date data
              REPLACE &f_name with ctod('  /  /  ')
         CASE type(f_name) = 'L'  && logical data
              REPLACE &f_name with .f.
         CASE type(f_name) = 'M'  && memo data
              REPLACE &f_name with ''
      ENDCASE
      i = i+1
   ENDDO
RETURN(rvalue)


FUNCTION add_rec
PRIVATE s_delete, rvalue
*** Function searches for a record deleted using
*** DELE_REC. if one is found, it return FALSE.
*** if not, the function append a record
*** to the data table and return TRUE.
IF len(alias()) = 0
   clear
   wait 'Function ADD_REC asked to ' +;
        'add a record to an open '+;
        'work area. Get serious!  '+;
        'Program aborting.' to junk
   CLOSE ALL
   CLEAR ALL
   CANCEL
ELSE
   s_delete = set('deleted') = 'ON'
   SET DELETED OFF          && set it off
   IF indexord() # 0  && if any index is active
      GO TOP                && go to top of file
      IF .not. deleted() && if no deleted record
         APPEND BLANK    && append a blank
         inputdata()
         rvalue = .t.
      ELSE                  && deleted record exists
        RECALL              && make it active again
        inputdata()
        rvalue = .f.
      ENDIF
   ELSE                     && no index active
      GO TOP                && go to top of file
      LOCATE FOR deleted()
      IF found()            && delete record found
         RECALL             && undelete the record
         inputdata()
         rvalue = .f.
      ELSE                  && if not found
         APPEND BLANK    && add a new record
         inputdata()
         rvalue = .t.
      ENDIF
   ENDIF
ENDIF
```

ได้ชัดด้วยการพิมพ์ box.ch มาดู จะเห็นว่าเป็นคำสั่งการกำหนดตัวอักษรกราฟิกของไอบีเอ็มเพื่อการตีกรอบ # define คือการกำหนดค่าคงที่ให้กับตัวแปร

ส่วนเมนโปรแกรม

เป็นโปรแกรมที่เริ่มต้นทำงานในที่นี้มีการกำหนดลูปให้ทำงาน โดยเรียกเมนูเป็นฟังก์ชัน ไวยากรณ์ที่ใช้คือ nchoice :=mainmenu ( ) นั่นคือ เมื่อเข้าไปทำงานในฟังก์ชัน mainmenu ( ) กลับมาจะได้ค่าส่งกลับมาให้ nchoice เพื่อแปลค่า nchoice ไปทำงานตามฟังก์ชัน listdata ( ) dele_rec ( ) และ add_rec ( )

ส่วนฟังก์ชัน

เป็นฟังก์ชันย่อย ๆ หลาย ๆ ฟังก์ชัน เหมือนภาษาซี เริ่มจากฟังก์ชัน mainmenu (menuchoice) มีพารามิเตอร์ menu-choice ที่จะส่งกลับ ดังนั้นปิดท้ายจึงใช้ return (menuchoice) คลิปเปอร์มีคำสั่ง prompt..message ให้สำหรับทำเมนู โดยค่าที่เลือกจะส่งกลับให้กับฟังก์ชันได้

- ฟังก์ชัน listdata เป็นฟังก์ชันที่เขียนขึ้นเพื่อ list ข้อมูลทุกเรคอร์ดออกทางจอภาพ

- ฟังก์ชัน dele_rec เป็นฟังก์ชันที่จะลบข้อมูลออกจากแฟ้ม โดยถามผู้ใช้ว่าจะใช้ลบข้อมูลเรคอร์ดใดออกจากแฟ้ม วิธีการลบจะทำเรคอร์ดนั้นให้ว่างโดยไม่ลบออกจริงแล้วมาร์กเรคอร์ดนั้นให้มี * กำกับไว้ ฟังก์ชัน dele_rec นี้เรียก rblank ( ) คือฟังก์ชันที่จะทำให้เรคอร์ดนั้นว่าง

- ฟังก์ชัน add_rec เป็นฟังก์ชันที่ใช้สำหรับการเพิ่มข้อมูลลงในแฟ้ม โดยการตรวจสอบว่ามีเรคอร์ดข่าวจาก

```
IF s_delete                &&  flag to be n
   SET DELETE on
ENDIF
RETURN(rvalue)

FUNCTION inputdata
         @ 5,25 clear to 20,80
         @ 3,28 ,19,65 box B_DOUBLE_SINGLE
         @ 5,40 say 'ADD NEW DATA'
         @ 6,32 say 'ID : ' get id picture 'xxxx'
         @ 8,32 say 'NAME : ' get name picture '@!'
         @ 10,32 say 'POSITION : ' get position picture '@!'
         @ 12,32 say 'BIRTH_DATE : 'get bdate picture '99/99/99'
         @ 14,32 say 'STATUS : ' get status picture '1'
         @ 16,32 say 'SALARY : ' get salary picture '99999999.99'
        read
RETURN .t.
```

การลบด้วยฟังก์ชัน dele_rec หรือไม่ ถ้ามีจะใส่ข้อมูลเพิ่มลงไปในเรคอร์ดนี้ แต่ถ้าไม่มีจะเพิ่มแบบต่อท้ายแฟ้ม (append blank) การเพิ่มข้อมูลนี้จึงมีการตรวจสอบแฟ้ม โดยดูว่าแฟ้มนี้เป็นแฟ้มดัชนีหรือไม่ โดยใช้ IF indexord ( ) # 0 ถ้าเป็นแฟ้มดัชนีจะหาเรคอร์ดข่าวที่ต้นแฟ้ม เพราะการลบทำให้เรคอร์ดข่าวจะไปปรากฏที่ต้นแฟ้ม แต่ถ้าไม่ใช่แฟ้มดัชนีจะใช้ locate for delete ( ) แทน ฟังก์ชัน add_rec มีการเรียก inputdata ( ) เพื่อป้อนข้อมูล

– ฟังก์ชัน inputdata ( ) คือ ฟังก์ชันสำหรับการป้อนข้อมูลเข้าไปในเรคอร์ดที่เปิดไว้ โดยโครงสร้างของแฟ้มนี้ประกอบด้วย id, name, position, birth_date, status และ salary รายละเอียดของโปรแกรมทั้งหมดแสดงได้ดังรูปที่ 4

ส่งท้าย

จากตัวอย่างโปรแกรมนี้มีจุดมุ่งหมายสองประการคือ ประการแรกต้องการให้เห็นโครงสร้างพื้นฐานของโปรแกรมที่เขียนด้วยคลิปเปอร์ 5.0 โดยเน้นการใช้คำสั่งและการคอมพายล์ตลอดจนการลิงค์

ประการที่สองต้องการแสดงตัวอย่างการลบเรคอร์ดโดยไม่ต้องใช้คำสั่ง pack แต่ทำเรคอร์ดให้ว่างไว้ เมื่อต้องการเพิ่มเรคอร์ดจะมาหาเรคอร์ดที่เคยลบไว้แล้วเพื่อกรอกข้อมูลได้ แต่ถ้าไม่มีเรคอร์ดที่ลบว่างอยู่ก็จะเพิ่มเรคอร์ดต่อท้าย ซึ่งเป็นหลักการที่เขียนไว้ในไมโครฉบับก่อน

คลิปเปอร์ 5.0 ยังมีจุดเด่นมากมาย ฉบับหน้าจะนำจุดเด่นต่าง ๆ มาเสนอให้ผู้สนใจได้เห็นถึงเทคนิคและพัฒนาการของระบบจัดการฐานข้อมูลซึ่งพัฒนาไปได้เร็วมาก

เมนู...เล่มล่า...

- การติดตั้งระบบ Config.sys และ Config.db 7
- จากดีเบสทรีไปเป็นดีเบสทรีพลัส 15
- การใช้ CATALOG 31
- การเขียนโปรแกรมด้วยดีเบสทรีพลัส 35
- เรื่องของตัวแปรและอะเรย์ 60
- ดีเบสกับเอ็มเอสดอส 70
- ความสัมพันธ์ระหว่างไฟล์ 77
- เวลาเป็นเงินเป็นทอง 87
- e^x log N และค่าทางตรีโกณมิติ 94
- ดีเบสกับการสร้างกราฟ 104
- ใช้ดีเบสทรีจัดเรียงคำภาษาไทย Thai Sort 122
- ดีเบสทรีพลัสกับภาษาแอสเซมบลี 131
- โปรแกรมจัดโครงสร้างโปรแกรม (Automatic Indent Lister: AIL version 1.01) 151

ฯลฯ

อิ่มเพียง 110 บาท

จาก

ไมโคร คอมพิวเตอร์